

EXHIBIT A

```
#!/internet/bin/perl5.002 -w
```

```
# Copyright (c) 1998
# Eugene Wang
```

```
# *** BEGIN ***
```

```
#-----
#-----
#input sequence (File 0) to compare
#-----
#-----
```

```
if ($#ARGV < 2) {die "argv < 2";}
open(EnzymeInput,$ARGV[0]) || die "Cannot open input file $ARGV[0]";
```

```
#print "Input Enzyme 1 sequence = ";
$E1sequence = <EnzymeInput>;
chomp $E1sequence;
$lenE1Seq = length($E1sequence);
$E1sequence =~ tr/a-z/A-Z/;
```

```
$E1ExtLoc = <EnzymeInput>;
chomp($E1ExtLoc);
$lenE1Total = $lenE1Seq + $E1ExtLoc;
```

```
#print "Input Enzyme 2 sequence = ";
$E2sequence = <EnzymeInput>;
chomp $E2sequence;
$E2sequence = reverse($E2sequence);
$lenE2Seq = length($E2sequence);
$E2sequence =~ tr/a-z/A-Z/;
```

```
$E2ExtLoc = <EnzymeInput>;
chomp($E2ExtLoc);
$lenE2Total = $lenE2Seq + $E2ExtLoc;
```

```
$lenE1Extra = $E2ExtLoc - $E1ExtLoc;
```

```
$E1SizeStart = <EnzymeInput>;
chomp($E1SizeStart);
$E1SizeEnd = <EnzymeInput>;
chomp($E1SizeEnd);
```

```
#-----
#-----
#open input FASTA file (File 1)
```



```

#-----
#-----
#print "Input file name = ";
#$fname = <>;
#chomp $fname;
#$fname = "H_DJ0167F23.seq";

open(Infile,$ARGV[1]) || die "Cannot open input file $ARGV[1]";

#-----
#-----
#open output file (File 2)
#-----

open (Outfile,">$ARGV[2]") || die "Cannot open output file $ARGV[2]";
#open (Outfile,">output.txt");
#print Outfile "Qualifier\tSequence";
#-----

#read input FASTA file
#-----

$line = <Infile>;      #header line
print Outfile "$line";
$linecount = 0;
$FullSeq = "";
#-----

#check headerline format
#-----

chomp $line;
@fields = split (/\/$/, $line);

$ntokens = 0;
foreach (@fields) {$ntokens++;}
#$ntokens = @fields;

if ($ntokens > 3)
    {$FragmentID = $fields[3];}
else
    {
        $line =~ s/^> />/;
        @fields = split (/ /, $line);
        $ntokens = 0;
        foreach (@fields) {$ntokens++;}
        if ($ntokens > 0)
            {$FragmentID = $fields[0]; $FragmentID =~ s/^> />/;}
        else
            {$FragmentID = "UnknownFragment";}
    }

while ($line = <Infile>)      #read in a line
    {

```

```

print Outfile "Enzyme top strand: ";
print Outfile "(5\'-$E1sequence";
if ($E1ExtLoc>0) {print Outfile "(N)$E1ExtLoc";}
print Outfile "-3\')";
print Outfile "\n";

print Outfile "Enzyme bottom strand: ";
print Outfile "(5\'-";
if ($E2ExtLoc>0) {print Outfile "(N)$E2ExtLoc";}
print Outfile "$E2sequence-3\')";
print Outfile " or ";
my $ts = reverse($E2sequence);
print Outfile "(3\'-$ts";
if ($E2ExtLoc>0) {print Outfile "(N)$E2ExtLoc";}
print Outfile "-5\')";

print Outfile "\n";

print Outfile "Segment size: $E1SizeStart - $E1SizeEnd\n";

$minLen = $lenE1Total < $lenE2Total ? $lenE1Total : $lenE2Total;
$maxLen = $lenE1Total > $lenE2Total ? $lenE1Total : $lenE2Total;

$nMatchE1 = 0;
$nSelected = 0;
@EnzLocLeft = ();
@EnzLocRight = ();
@EnzTypeLeft = ();
@EnzTypeRight = ();

if ($minLen > 0)
{
#   for ($i=0; $i <= $lenFullSeq-$lenE1Seq; $i++)
#   for ($i=0; $i <= $lenFullSeq-$maxLen; $i++)
#   {
#       if (substr($FullSeq,$i,$lenE1Seq) eq $E1sequence)
#       {
#           $EnzLocLeft[$nMatchE1] = $i + $lenE1Total;
##have to use push()
#           $EnzTypeLeft[$nMatchE1] = 1;
#           push(@EnzLocLeft,$i + $lenE1Total);
#           push(@EnzTypeLeft,1);

#           print Outfile "$nMatchE1\t$i\t";
#           print Outfile "type 1\t";
#           print Outfile "$E1sequence\t";
#           print Outfile substr($FullSeq,$i,$lenE1Total);
#           print Outfile "\n";

#           if ($nMatchE1 > 0)
#           {
#               push(@EnzLocRight,$i + $lenE1Total-1);
#               push(@EnzTypeRight,1);
#           }

```

```

        $nMatchE1++;
    }
#       if (substr($FullSeq,$i+$E2ExtLoc,$lenE2Seq) eq
$E2sequence)
#       elseif (substr($FullSeq,$i+$E2ExtLoc,$lenE2Seq) eq
$E2sequence)
#       {
#           $EnzLocLeft[$nMatchE1] = $i;
#           $EnzCutLeft[$nMatchE1] = 2;
#           push(@EnzLocLeft,$i);
#           push(@EnzTypeLeft,2);
#
#           print Outfile "$nMatchE1\t$i\t";
#           print Outfile "type 2\t";
#           print Outfile "$E2sequence\t";
#           print Outfile substr($FullSeq,$i,$lenE2Total);
#           print Outfile "\n";
#
#           if ($nMatchE1 > 0)
#           {
#               push(@EnzLocRight,$i-1);
#               push(@EnzTypeRight,2);
#           }
#
#           $nMatchE1++;
#       }
    }

    if ($nMatchE1 > 0)
    {
        push(@EnzLocRight,$i-1);
        push(@EnzTypeRight,2);
    }

    print Outfile "Number of segments: $nMatchE1\n";
    if ($nMatchE1 != ($#EnzLocRight+1)) {die ("Counting
error...nMatchE1($nMatchE1) != $#EnzLocRight");}

    print Outfile "Matched loci:\n";

    for ($i=0; $i < $nMatchE1; $i++)
    {
        print Outfile "$EnzLocLeft[$i]\t";
    }

    print Outfile "\nSegment Size:\n";
    for ($i=0; $i < $nMatchE1-1; $i++)
    {
        $tmpSegSize = $EnzLocRight[$i] - $EnzLocLeft[$i] + 1;
        if ($tmpSegSize >= $E1SizeStart && $tmpSegSize <=
$E1SizeEnd)
        {
            $SegSelected[$nSelected++] = $i;

```

```

    }
    print Outfile "$tmpSegSize\t";
  }
}

##-----
##      print out the Segment (E1) sequences
##-----
-----
print Outfile "\nSegments Selected ($nSelected):";
for ($i=0; $i < $nSelected; $i++)
{
  $selSeq = $SegSelected[$i];
  $Elleft = $EnzLocLeft[$selSeq];
  $Elright = $EnzLocRight[$selSeq];

  if ($lenElExtra > 0) {$Elright += $lenElExtra;}
  else {$Elleft += $lenElExtra;}
  $lenSelSeq = $Elright - $Elleft + 1;

  $OutputHeaderLine = ">" . $FragmentID . "_" . $selSeq . "\tsize=" .
  $lenSelSeq;
  $OutputHeaderLine .= "\tLocI=" . $Elleft . "-" . $Elright;
  $OutputHeaderLine .= "\tEnz$EnzTypeLeft[$selSeq]-
  Enz$EnzTypeRight[$selSeq]";

  print Outfile "\n$OutputHeaderLine";
  print "$OutputHeaderLine";

  #      Segment sequence
  $SeqEltoNextEl = substr($FullSeq,$Elleft,$lenSelSeq);
  print Outfile "\n$SeqEltoNextEl\n";
  print "\n$SeqEltoNextEl\n";

}

return ($lenFullSeq);
}

```

EXHIBIT B

```
#!/internet/bin/perl5.002 -w

#*****
# Copyright (c) 1998
# Author: Eugene Wang
# Title: Ligate
# Purpose: Find matching segments/sequences in two files
#*****
if ($#ARGV != 2) {die "Number of argv ($#ARGV+1) != 3";}

#-----
#input file
#-----

open(InfileLigate,$ARGV[0]) or die "Open error...$ARGV[0]\n";

$locLigate = <InfileLigate>;
chomp $locLigate;
$seqLigate = <InfileLigate>;
chomp $seqLigate;

close (InfileLigate);
#-----

#output file
#-----

open(Infile,$ARGV[1]) or die "Open error...$ARGV[1]\n";

$OutName = $ARGV[2];
open (Outfile,">$OutName") or die("Open error...$OutName");

$alreadyReadOne = 0;
$sequence = "";
while ($line = <Infile>)      #read in a line
{
    chomp $line;
    next if ($line eq "");
    if ($line =~ /^#/ || $line =~ /^>/)      ##if first char is a '#'
        or '>'
    {
        if ($alreadyReadOne == 1) {
            if (&Ligate($sequence,$locLigate,$seqLigate) == 1) {
                print Outfile "$headerLine\n";
                print Outfile "$sequence\n";
            };
            $sequence = "";
        }
    }
}
```

```

        $headerLine = $line;
        $alreadyReadOne = 1;
    }
    else
    {
        $sequence .= $line;
    }
}

if ($alreadyReadOne == 1) {
    if (&Ligate($sequence,$locLigate,$seqLigate) == 1) {
        print Outfile "$headerLine\n";
        print Outfile "$sequence\n";
    };
}

close (Infile);
close (Outfile);

#####
#####
#compare sequence with Ligation Adapter sequence
#####
#####
sub Ligate()
{
    local $retcode = 0;

    local ($seq,$locLigate,$seqLigate) = @_ ;

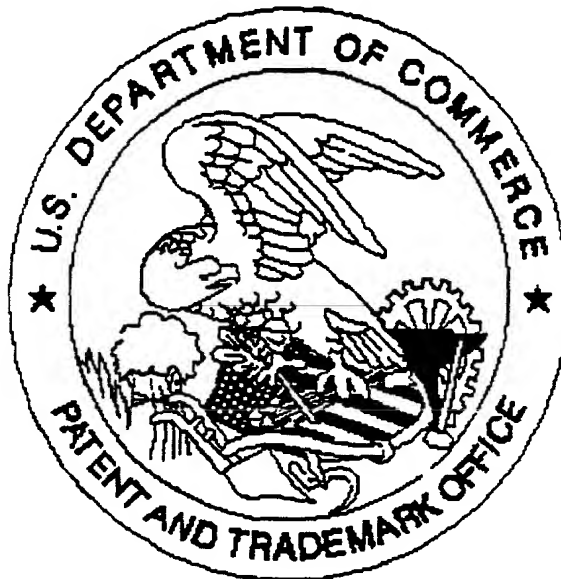
    local $lenLigate = length($seqLigate);
    local $lenSeq = length($seq);

    if ((substr($seq,$locLigate,$lenLigate) eq $seqLigate) &&
        (substr($seq,$lenSeq-$locLigate-$lenLigate,$lenLigate) eq
        $seqLigate)) {
        $retcode = 1;
    }

    return $retcode;
}

```

United States Patent & Trademark Office
Office of Initial Patent Examination -- Scanning Division



Application deficiencies found during scanning:

☐ Page(s) _____ of _____ were not present
for scanning. (Document title)

☐ Page(s) _____ of _____ were not present
for scanning. (Document title)

There are only 19 pages of drawing.

☒ *Scanned copy is best available. Drawings*